Project discussion, 22 May: Mandatory but ungraded.

Thanks for doing this

June 4, 6pm deadline for submitting poster for printing (pdf preferred). TAs have to print 43 posters. Dropbox link or email TA
https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

June 5, 5-8pm Atkinson Hall: Poster and Pizza. Easels available.

June 15, deadline for submitting report and code. (we have 43 reports to read in 3 days!) use dropbox link or email TA
https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

Evaluation
Report:30%
Poster: 10% (as displayed)
Code: 10% (should run automatically)
Generative Models

Given training data, generate new samples from same distribution

Training data $\sim p_{data}(x)$
Generated samples $\sim p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

**Several flavors:**
- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it
Taxonomy of Generative Models

Today: discuss 3 most popular types of generative models today

- Explicit density
  - Fully Visible Belief Nets
    - NADE
    - MADE
    - PixelRNN/CNN
  - Change of variables models (nonlinear ICA)

- Implicit density
  - Approximate density
    - Variational
  - Markov Chain
    - Variational Autoencoder
    - Boltzmann Machine

Generative models

Direct
- GAN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
Some background first: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function: $\| x - \hat{x} \|^2$

Doesn’t use labels!

After training, throw away decoder
**Variational Bayes summary**

Bayes \( p(x|y) = \frac{p(y|x)p(y)}{p(x)} \)

Optimizing posterior \( p(x|y) \)

You can also optimize the evidence (type II likelihood) \( p(y) \)

------

Observations \( X = [x_1, ..., x_N] \)
With latent parameter \( Z = [z_1, ..., z_N] \)
And probability \( p(X, Z) \)
We like to find an approximation to \( p(X, Z) \) and the evidence \( p(Z) \)
A good guess is a factorized distribution
\( p(X, Z) = \prod_{n=1}^{N} p(z_n) \)

Bishop Ch 10 Approximate inference
Variational inference
So far...

PixelCNNs define tractable density function, optimize likelihood of training data:
\[
p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, \ldots, x_{i-1})
\]

VAEs define intractable density function with latent \( z \):
\[
p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz
\]

Cannot optimize directly, derive and optimize lower bound on likelihood instead.

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don’t work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game
Implicit Deep Generative Modeling

\[ z \sim N(0, I) \]  
\[ W_1 x_1 + b_1 \]  
\[ W_2 x_2 + b_2 \]  
\[ \vdots \]  
\[ W_t x_t + b_t \]  
\[ x \sim p_\theta(x) \approx p_{gt}(x) \]

\( \Rightarrow \) easy to sample

Example

Popular Example

Generative Adversarial Networks:

- Based on game theory, Nash equilibrium
- 8679 citations [Goodfellow et al., 2014]
Generative Adversarial Networks (GANs)

\[
\{z^{(j)}\}_{j=1}^m, \quad z^{(j)} \sim N(z | 0, I)
\]

\[
\{x^{(i)}\}_{i=1}^n, \quad x^{(i)} \in \mathbb{R}^d \sim p_{gt}(x)
\]

\[
\begin{align*}
D_\phi : \mathbb{R}^d &\rightarrow (0, 1) \\
D_\phi(x^{(i)}) &\approx 1, \quad D_\phi(G_\theta(z^{(j)})) \approx 0
\end{align*}
\]

\[
\begin{align*}
\text{Binary classification:} \\
\text{real} &\quad \text{fake}
\end{align*}
\]

Basic GAN objective (cross-entropy-based):

\[
\min_{\theta} \max_{\phi} \quad \mathbb{E}_{p_{gt}(x)} \left[ \log D_\phi (x) \right] + \mathbb{E}_{N(z | 0, I)} \left[ \log \left( 1 - D_\phi \left[ G_\theta (z) \right] \right) \right]
\]

[Goodfellow et al., 2014]
GAN Strengths

State-of-the-art GAN models generate highly realistic samples, e.g., StyleGAN [Karras et al, 2019]:

Examples from http://www.whichfaceisreal.com/
GAN Weaknesses

- Training involves potentially unstable minimax problem, iterations may diverge, be sensitive to tuning.  
  \cite{Lucic2018}

- Can be susceptible to mode collapse:

- No explicit density estimate $p_0(x) \approx p_{gt}(x)$, cannot infer the latent code that produced a sample: $p_0(z \mid x)$?

\[ \text{low sample diversity} \]

\[ \text{cannot compute low-dimensional representation} \]
**Comparison with an Autoencoder**

**Autoencoder (AE):**
- $x$ input
- **Encoder** DNN
- $z$ compressed representation, $\text{dim}(z) \ll \text{dim}(x)$
- **Decoder** DNN
- $\hat{x}$ output

**VAE:**
- $x$ input
- **Encoder** DNN
- $\mu_z$, $\Sigma_z$ stochastic representation
- $z \sim q_\phi(z \mid x) = N(\mu_z, \Sigma_z)$
- **Decoder** DNN
- $\mu_x$, $\Sigma_x$
- $\hat{x} \sim p_\theta(x \mid z) = N(\mu_x, \Sigma_x)$
- $\hat{x}$ output
Variational Autoencoders
(details in Part II)

Advantages:

- Less prone to mode collapse than GANs, more stable training.
- Provides explicit estimate of latent distribution $p_0(z|x)$; many applications in representation learning.
- Natural generalization of dimensionality reduction tools in common use for signal processing (Part III).

Disadvantages:

- Optimizes a bound on the data likelihood, not exact likelihood (but conditions for when bound is tight discussed in Part IV).
- Generated samples usually inferior to GANs ...

... although improvements possible (Part IV).
Representative Applications

Generative models in general:

- Model-based reinforcement learning:
  - Compression:
    - Encoder
    - DNN
    - \( x \rightarrow \hat{x} \) 
    - \( z \) compressible representation
  - Data cleaning, outlier removal:
    - \[ \text{[Dai et al., 2018]} \]
    - \[ \text{[Ballé et al., 2018]} \]

- Image-to-image translation:
  - \[ \text{[Isola et al., 2016]} \]

- Many more, a generic unsupervised learning tool

VAEs in particular:

- Compression:
  - \[ \text{[Finn et al., 2016]} \]

- Data cleaning, outlier removal:
State space model

\[ X_0 \sim N(0, P_0) \]

\[ X_1 \rightarrow X_2 \]

\[ y_1 \rightarrow y_2 \]

State Eq.
\[ X_{R+1} = M_R X_R + \xi_R \]

Measurement Eq.
\[ y_R = H_R X_R + V_R \]

\[ \xi_R \sim N(0, Q) \]
\[ V_R \sim N(0, R) \]
The Model

Consider the discrete, linear system,

\[ x_{k+1} = M_k x_k + w_k, \quad k = 0, 1, 2, \ldots, \]  

(1)

where

- \( x_k \in \mathbb{R}^n \) is the state vector at time \( t_k \)
- \( M_k \in \mathbb{R}^{n \times n} \) is the state transition matrix (mapping from time \( t_k \) to \( t_{k+1} \)) or model
- \( \{w_k \in \mathbb{R}^n; k = 0, 1, 2, \ldots\} \) is a white, Gaussian sequence, with \( w_k \sim \mathcal{N}(0, Q_k) \), often referred to as model error
- \( Q_k \in \mathbb{R}^{n \times n} \) is a symmetric positive definite covariance matrix (known as the model error covariance matrix).

Some of the following slides are from: Sarah Dance, University of Reading
The Observations

We also have discrete, linear observations that satisfy

$$y_k = H_k x_k + v_k, \quad k = 1, 2, 3, \ldots, \quad (2)$$

where

- $y_k \in \mathbb{R}^p$ is the vector of actual measurements or observations at time $t_k$.
- $H_k \in \mathbb{R}^{n \times p}$ is the observation operator. Note that this is not in general a square matrix.
- $\{v_k \in \mathbb{R}^p; k = 1, 2, \ldots\}$ is a white, Gaussian sequence, with $v_k \sim \mathcal{N}(0, R_k)$, often referred to as observation error.
- $R_k \in \mathbb{R}^{p \times p}$ is a symmetric positive definite covariance matrix (known as the observation error covariance matrix).

We assume that the initial state, $x_0$ and the noise vectors at each step, $\{w_k\}$, $\{v_k\}$, are assumed mutually independent.
Summary of the Kalman filter

**Prediction step**
- **Mean update:**
  \[ \hat{x}_{k+1|k} = M_k \hat{x}_{k|k} \]
  \[ P_{k+1|k} = M_k P_{k|k} M_k^T + Q_k. \]

**Observation update step**
- **Mean update:**
  \[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - H_k \hat{x}_{k|k-1}) \]
  \[ K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \]
  \[ P_{k|k} = (I - K_k H_k) P_{k|k-1}. \]
Given

\[ p(x) = \mathcal{N}(x|\mu, \Lambda^{-1}) \]

we have

\[ p(y|x) = \mathcal{N}(y|Ax + b, L^{-1}) \]

\[ p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T) \]

\[ p(x|y) = \mathcal{N}(x|\Sigma\{A^TL(y - b) + \Lambda\mu\}, \Sigma) \]

where

\[ \Sigma = (\Lambda + A^TLA)^{-1} \]
Simultaneous location and mapping (SLAM)

Graphical model underlying SLAM. \( L^i \) is the fixed location of landmark \( i \), \( x_t \) is the robot location, and \( y_t \) is the observation. In this trace, the robot sees landmarks 1 and 2 at time 1, then just landmark 2, then just landmark 1, etc.

Illustration of the SLAM problem. (a) A robot starts at the top left and moves clockwise in a circle back to where it started. We see how the posterior uncertainty about the robot’s location increases and then decreases as it returns to a familiar location, closing the loop. If we performed smoothing, this new information would propagate backwards in time to disambiguate the entire trajectory.
Constant velocity model

Using a constant velocity CV track model for the source, the state equation is given by

$$x_{k+1} = \begin{bmatrix} d_{k+1} \\ v_{k+1} \end{bmatrix} = M_k x_k + B_k \varepsilon_k = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_k \\ v_k \end{bmatrix} + \begin{bmatrix} 1/2 \Delta^2 \\ 1 \end{bmatrix} \varepsilon_k$$

where

$$\Delta = t_{k+1} - t_k$$

Note that the noise term on velocity is now an acceleration in the location-term.
Figure 17.11: The main kinds of inference for state-space models. The shaded region is the interval for which we have data. The arrow represents the time step at which we want to perform inference. $t$ is the current time, $T$ is the sequence length, $\ell$ is the lag and $h$ is the prediction horizon. See text for details.

- **Fixed lag smoothing**
  - This involves computing $p(z_t - \ell | x_1: t)$, where $\ell > 0$ is called the lag. This gives better performance than filtering, but incurs a slight delay. By changing the size of the lag, one can trade off accuracy vs delay.

- **Prediction**
  - Instead of predicting the past given the future, as in fixed lag smoothing, we might want to predict the future given the past, i.e., to compute $p(z_{t+h} | x_1: t)$, where $h > 0$ is called the prediction horizon. For example, suppose $h = 2$; then we have $p(z_{t+2} | x_1: t) = \sum_{z_{t+1}} p(z_{t+2} | z_{t+1}) p(z_{t+1} | x_1: t)$ (17.42). It is straightforward to perform this computation: we just power up the transition matrix and apply it to the current belief state. The quantity $p(z_{t+h} | x_1: t)$ is a prediction about future hidden states; it can be converted into a prediction about future observations using $p(x_{t+h} | x_1: t) = \sum_{z_{t+h}} p(x_{t+h} | z_{t+h}) p(z_{t+h} | x_1: t)$ (17.43). This is the posterior predictive density, and can be used for time-series forecasting (see Fraser 2008 for details). See Figure 17.11 for a sketch of the relationship between filtering, smoothing, and prediction.

- **MAP estimation**
  - This means computing $\arg \max_{z_{1:T}} p(z_{1:T} | x_1:T)$, which is a possible state sequence. In the context of HMMs, this is known as Viterbi decoding.
Kalman smoother

Figure 18.1 Kalman filtering and smoothing. (a) Observations (green circles) are generated by an object moving to the right (true location denoted by black squares). (b) Filtered estimated is shown by dotted red line. Red cross is the posterior mean, blue circles are 95% confidence ellipses derived from the posterior covariance. For clarity, we only plot the ellipses every other time step. (c) Same as (b), but using offline Kalman smoothing. Figure generated by kalmanTrackingDemo.
Predict N steps ahead
SLAM (Simultaneous Location and Mapping)
Kalman smoother
RLS (Recursive least squares)

Advanced KF:
- Ensample KF (EnKF) non Gaussian
- Extended KF (EKF) non-linear
- Unscented KF (UKF) well chosen control points
- ... Particle Filter Nonlinear,
- .... non Gaussian

Advanced KF:
- Ensample KF (EnKF) non Gaussian
- Extended KF (EKF) non-linear
- Unscented KF (UKF) well chosen control points
- ... Particle Filter Nonlinear,
- .... non Gaussian

Advanced KF:
- Ensample KF (EnKF) non Gaussian
- Extended KF (EKF) non-linear
- Unscented KF (UKF) well chosen control points
- ... Particle Filter Nonlinear,
- .... non Gaussian

Advanced KF:
- Ensample KF (EnKF) non Gaussian
- Extended KF (EKF) non-linear
- Unscented KF (UKF) well chosen control points
- ... Particle Filter Nonlinear,
- .... non Gaussian
**Linear regression:** Linear Basis Function Models (1)

Generally

\[ y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x) \]

- where \( \phi_j(x) \) are known as **basis functions**.
- Typically, \( \phi_0(x) = 1 \), so that \( w_0 \) acts as a bias.
- Simplest case is linear basis functions: \( \phi_d(x) = x_d \).

\[ y = \phi^T \cdot x \]
Sparse model

Model: \( y = Ax + n \), \( x \) is sparse

- \( y \): measurements
- \( n \): noise
- \( x \): sparse weights
- Dictionary (A) – either from physical models or learned from data (dictionary learning)

\( N \times 1 \) measurements

\( N \times M \)

\( M \times 1 \) sparse signal
Sparse processing

- Linear regression (with sparsity constraints)
  - An underdetermined system of equations has many solutions
  - Utilizing x is sparse it can often be solved
  - This depends on the structure of A (RIP – Restricted Isometry Property)

- Various sparse algorithms
  - Convex optimization (Basis pursuit / LASSO / L₁ regularization)
  - Greedy search (Matching pursuit / OMP)
  - Bayesian analysis (Sparse Bayesian learning / SBL)

- Low-dimensional understanding of high-dimensional data sets

- Also referred to as compressive sensing (CS)
Different applications, but the same algorithm

Model: \( y = Ax + n \), \( x \) is sparse

<table>
<thead>
<tr>
<th>( y )</th>
<th>( A )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency signal</td>
<td>DFT matrix</td>
<td>Time-signal</td>
</tr>
<tr>
<td>Compressed-Image</td>
<td>Random matrix</td>
<td>Pixel-image</td>
</tr>
<tr>
<td>Array signals</td>
<td>Beam weight</td>
<td>Source-location</td>
</tr>
<tr>
<td>Reflection sequence</td>
<td>Time delay</td>
<td>Layer-reflector</td>
</tr>
</tbody>
</table>
CS approach to geophysical data analysis

CS of Earthquakes
Yao, GRL 2011, PNAS 2013

Sequential CS
Mecklenbrauker, TSP 2013

CS beamforming
Xenaki, JASA 2014, 2015
Gerstoft JASA 2015

CS fathometer
Yardim, JASA 2014

CS Sound speed estimation
Bianco, JASA 2016

CS matched field
Gemba, JASA 2016
Sparse signals /compressive signals are important

- We don’t need to sample at the Nyquist rate

- Many signals are sparse, but are solved them under non-sparse assumptions
  - Beamforming
  - Fourier transform
  - Layered structure

- Inverse methods are inherently sparse: We seek the simplest way to describe the data

- All this requires new developments
  - Mathematical theory
  - New algorithms (interior point solvers, convex optimization)
  - Signal processing
  - New applications/demonstrations
Sparse Recovery

• We try to find the sparsest solution which explains our noisy measurements
• $L_0$-norm

Here, the $L_0$-norm is a shorthand notation for counting the number of non-zero elements in $x$. 
Sparse Recovery using $L_0$-norm

Underdetermined problem

$$y = Ax, \quad M < N$$

Prior information

$x$: $K$-sparse, $K \ll N$

There are only few sources with unknown locations and amplitudes

- $L_0$-norm solution involves exhaustive search
- Combinatorial complexity, not computationally feasible
L_p-norm

\[ \| x \|_p = \left( \sum_{m=1}^{M} |x_m|^p \right)^{1/p} \quad \text{for } p > 0 \]

- Classic choices for \( p \) are 1, 2, and \( \infty \).
- We will misuse notation and allow also \( p = 0 \).
\[ \|x\|_p = \left( \sum_{m=1}^{M} |x_m|^p \right)^{1/p} \]
Solutions for sparse recovery

• Exhaustive search
  – $L_0$ regularization, not computationally feasible

• Convex optimization
  – Basis pursuit / LASSO / $L_1$ regularization

• Greedy search
  – Matching pursuit / Orthogonal matching pursuit (OMP)

• Bayesian analysis
  – Sparse Bayesian Learning / SBL

• Regularized least squares
  – $L_2$ regularization, reference solution, not actually sparse
Regularized least squares

\[ \tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} \|w\|^2 \]

The squared weights penalty is mathematically compatible with the squared error function, giving a closed form for the optimal weights:

\[ w^* = (\lambda I + X^T X)^{-1} X^T t \]

A picture of the effect of the regularizer

- The overall cost function is the sum of two parabolic bowls.
- The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The L2 regularizer just shrinks the weights.
Basis Pursuit / LASSO / $L_1$ regularization

- The $L_0$-norm minimization is not convex and requires combinatorial search making it computationally impractical

- We make the problem convex by substituting the $L_1$-norm in place of the $L_0$-norm

$$\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 < \varepsilon$$

- This can also be formulated as

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|x\|_1 < \varepsilon$$

$$\min \|Ax - b\|_2 + \lambda (\|x\|_2 - \varepsilon) = \min \|Ax - b\|_2 + \lambda \|x\|_2$$
The unconstrained -LASSO- formulation

Constrained formulation of the $\ell_1$-norm minimization problem:

$$\hat{x}_{\ell_1}(\epsilon) = \arg\min_{x \in \mathbb{C}^N} \|x\|_1 \text{ subject to } \|y - Ax\|_2 \leq \epsilon$$

Unconstrained formulation in the form of least squares optimization with an $\ell_1$-norm regularizer:

$$\hat{x}_{\text{LASSO}}(\mu) = \arg\min_{x \in \mathbb{C}^N} \|y - Ax\|_2^2 + \mu \|x\|_1$$

For every $\epsilon$ exists a $\mu$ so that the two formulations are equivalent

Regularization parameter : $\mu$
Basis Pursuit / LASSO / $L_1$ regularization

- Why is it OK to substitute the $L_1$-norm for the $L_0$-norm?

- What are the conditions such that the two problems have the same solution?

\[
\begin{align*}
\min_x \| x \|_1 & \quad \text{subject to } \| Ax - b \|_2 < \varepsilon \\
\min_x \| x \|_0 & \quad \text{subject to } \| Ax - b \|_2 < \varepsilon
\end{align*}
\]

- Restricted Isometry Property (RIP)

\[
(1 - \delta_s) \| u \|_2 \leq \| A_s u \|_2 \leq (1 + \delta_s) \| u \|_2
\]
Geometrical view of the lasso compared with a penalty on the squared weights

L_2 regularization

L_1 regularization
Regularization parameter selection

The objective function of the LASSO problem:

$$L(x, \mu) = \|y - Ax\|_{2}^{2} + \mu \|x\|_{1}$$

- Regularization parameter : $\mu$
- Sparsity depends on $\mu$
  - $\mu$ large, $x = 0$
  - $\mu$ small, non-sparse

We can predict the jump in support
• As regularization parameter $\mu$ is decreased, more and more weights become active

• Thus $\mu$ controls sparsity of solutions
Applications

- MEG/EEG/MRI source location (earthquake location)
- Channel equalization
- Compressive sampling (beyond Nyquist sampling)
- Compressive camera!
- Beamforming
- Fathometer
- Geoacoustic inversion
- Sequential estimation